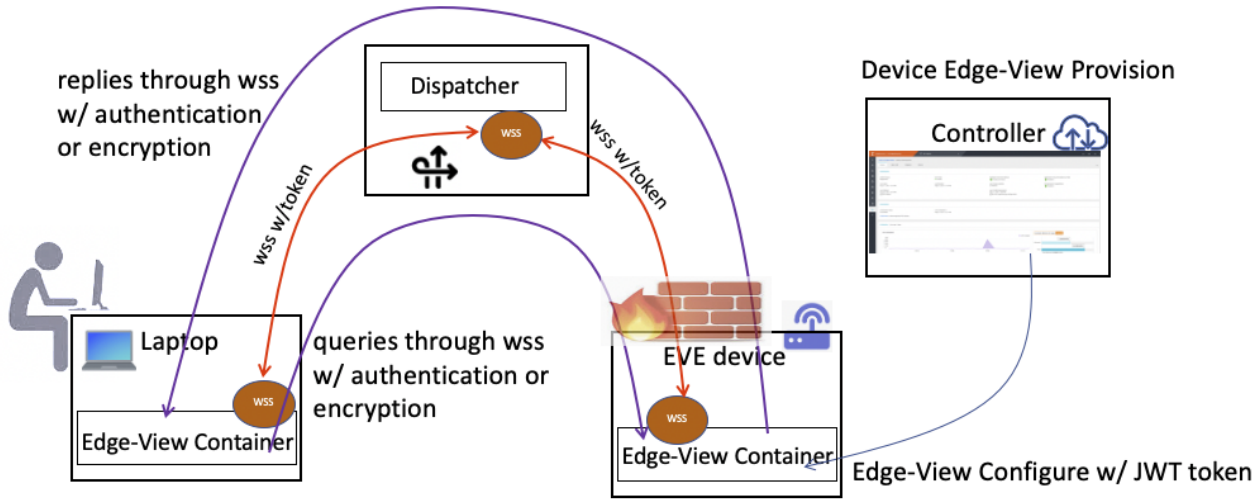
Edge-View Architecture

Introduction

Edge-View is a tool to allow remote operators or engineers to securely access the edge-nodes for device and application debugging and/or performing operation. The edge-nodes may be behind a firewall, NAT, LTE or proxy server.



There are four items in the figure:

1. controller
2. edge-node
3. dispatcher
4. user laptop.

Controller is to define the policy, generate JWT (JSON Web Token) with a timeout value to be downloaded to edge-node as part of the device configuration; edge-node receives the configuration on Edge-View and sets up the connection to the Dispatcher and ready to receive requests from the user; The Dispatcher runs in the cloud usually and handles the distribution of edge-view packets between the edge-node and the user; The laptop also runs Edge-View container and is used to query into the edge-node or to setup TCP connections into the edge-node.

This document describes the end-to-end operation, provisioning and status update to the controller of Edge-View. How the Edge-View works and its features is outside the scope of this document.

The edge-view container code will be published in 'github.com/lf-edge/eve/pkg/edgeview'. The 'edge-view' containers run on the EVE device and on the laptop are basically the same container with different wrappers.

In the case of provisioning, an enterprise user can go into the device section of the controller, and click a button to enable the 'edge-view' on the device and the default timeout duration. A JWT token will be displayed on the page, and the user can copy this down and send this token to someone who can perform the troubleshooting of the issues on the device or the applications on the device.

All the messages between the laptop and the EVE device is either authenticated or encrypted using the JWT nouce, and the websocket session through the dispatcher is TLS based.

Laptop

The Edge-View container client normally runs on the operator's laptop. Different query/debug commands can be issued on the laptop, and get the results from the remote edge-node through the 'dispatcher' virtual connection.

To build the Edge-View container yourself, cd to 'github.com/lf-edge/eve/pkg/edgeview' and do a 'make edge-view-query' and it will build the 'edge-view-query' docker image.

The laptop can also use the Edge-View container in 'ssh-mode', which needs Internet access to the edge-node without the firewall/NAT/LTE/proxy in the middle. The 'ssh-mode' does not involve the 'dispatcher' and controller pieces. This document will only concern the 'non-ssh-mode' operation.

EVE Device

When the Eve image containing the Edge-View and the container is running, there is a script in the loop that monitors the configuration status. If the controller sends configuration to the edge-node with edge-view enabled and the allowed timestamp is valid, it will launch the connection towards the defined Dispatcher endpoint and be ready for the queries from the user side. The edge-node also enforces the policies set by the configuration for those queries. For instance, it checks to see if the ssh into the edge-node is allowed through Edge-View connection or not.

For now, to build an EVE image with the Edge-View container, in your 'github.com/lf-edge/eve' workspace, do a 'make HV=edgeview-kvm rootfs'.

Controller

Provisioning

To enable the Edge-View on an edge-node, the edge-node configuration needs to include the Edge-View specific configure. It includes a JWT token string, the policies for edge-node and application, etc.

Config API

Below is the API for protobuf definition as part of the EdgeDevConfig:

```
message EdgeViewConfig {
  // JWT token for signed info, it contains the dispatcher
  // endpoint IP:Port, device UUID, nonce and expiration time
  string  token = 1;
  // dispatcher certificate(s) if it's not well-known CA signed
  repeated bytes disp_cert_pem = 2;
  // policy for device access through edge-view
  DevDebugAccessPolicy dev_policy = 3;
  // policy access for apps through edge-view
  AppDebugAccessPolicy app_policy = 4;
}

// Dev debug policy applicable to edge-view
message DevDebugAccessPolicy {
  // device side of edge-view access is allowed or not
  bool allow_dev = 1;
}

// App debug policy applicable to edge-view
message AppDebugAccessPolicy {
  // app side of edge-view access is allowed or not
  bool allow_app = 1;
}
```

JWT Token

The 'token' in the configure message is a JWT string, which contains three sections 1) the header or algorithm 2) the Edge-View configure data 3) the controller signature of the token

The algorithm for signature is SHA256withECDSA and with the same signing certificate as in the V2 API envelope by controller. Both the edge-node and the user laptop side can get the controller signing public certificate and verify the JWT token.

The configuration part is in JSON format as defined below:

```
type EvjwtInfo struct {
  Dep string `json:"dep"`     // dispatcher end-point string e.g. ip:port
  Sub string `json:"sub"`     // jwt subject, the device UUID string
  Exp uint64 `json:"exp"`     // expiration time for the token
  Key string `json:"key"`     // key or nonce for payload hmac authentication
  Num uint8  `json:"num"` // number of instances, default is 1
  Enc bool   `json:"enc"`     // payload with encryption, default is authentication
}
```

- The “Exp” is the expiration timestamp in Unix format in seconds. The edge-node will shut the Edge-View container if the time has expired.
- The “Sub” is the device UUID, which defines that only the intended edge-node can use this JWT token.
- The “Dep” is the dispatcher end-point URL, which includes the IP address or domain name, the TCP port and the path.
- The “Key” is the nonce of either authentication or encryption operation for the Edge-view messages between the user laptop and the device.
- The “Num” is the number of Edge-view instances running on the device. It can support up to 5 Edge-view instances to allow multiple sessions on the device simultaneously.
- The boolean “Enc” is for the encryption operation of the message, the default is authentication.

Below is an example of a JWT token generated for an edge-node(sc-supermicro-zc5), it expires on April 30, 2022 with some example dispatcher endpoint (the purple color marks the 2nd part of JWT, the payload part):

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJkZXAiOiIzNC4xMjEuMTAwLjIwMDozODM4L2VkZ2UtdmlldyIsImVuYyI6ZmFsc2UsImV4cCI6MTY1MTM2NTY0MCwia2V5Ijoic29tZS1yYW5kb20tc3RyaW5nIiwibnVtIjozLCJzdWIiOiI1ZmY0YmY3Zi0xZjdkLTRiMGYtODg2My0wMDQ0YzcyYWMyZGYifQ.
22CrDTcFPwtB5zDj7ouw_9P3DvulokAmYL8Sx7wU6-wwK5_uKHD-7KFsFM9sEXLc7z1MfV1_YQ0f-QXhlL-Lpw

- JWT token len 322: {Alg:ES256 Typ:JWT}
- token: {Dep:34.121.100.200:3838/edge-view Sub:5ff4bf7f-1f7d-4b0f-8863-0044c72ac2df Exp:1651365640 Key:some-random-string Num:3 Enc: false}
- expires: 2022-04-30 17:40:40 -0700 PDT

Assume the ‘Dep’ for an enterprise is defined and inherited, the controller knows the UUID of the edge-node which is being provisioned, the only item in the JWT may need to change is the ‘Exp’ if the user does not want to use the default timeout value. E.g. the user wants this JWT to be valid only for the next 6 hours. The above JWT defines multiple instances of 3 and uses nonce authentication mechanism.

Policies for edge-node and application

The policy may be defined in 3 levels. The enterprise, the project and the edge-node. The lower level can inherit the definition of the upper level to be scalable.

Initially, the API is defined to only allow two top level permissions, the device and application. We need to add more detail when we get some experience on the Edge-view usages.

Dispatcher certificates

This is an optional configure and it is there just in case if a customer wants to own and manage their Edge-View dispatcher with certificates using private CA.

Suppress the Edge-View Configuration

When the edge-node configuration is being downloaded, it is optional to check if it’s JWT token has expired or not. It may not include the Edge-View configuration part if it has expired since the edge-node can not use that anyway. This is only from reducing the configure size point of view as an optimization.

Use of the JWT

The JWT token string is the only piece of information for the Edge-View container (besides the policy configuration on the edge-node) to communicate between the laptop and edge-node. This JWT token string may need to be displayed on the edge-node UI page. This token string can be attached to e.g. an email for the intended operator/engineer to use on his/her laptop or on a debug machine. There should be a decode action button on the UI to allow the user to see the contents of the token (there is no secret in it).

On the user laptop, the operator needs to use the token when issuing commands e.g.:

‘docker … edge-view-query ***-token <jwt-string>*** [-inst <1-5>] <query-cmd>’

The operator does not need to know explicitly where the dispatcher is, which edge-node he/she will connect to or where the edge-node is located.

Edge-View can have multiple instances running on the same device, defined in JWT ‘Num’ value larger than 1. The user launching the ‘edge-view-query’ on a laptop needs to specify the instance number as an input.

Post Provisioning

After the controller provisioning of the Edge-View to the edge-node. The Edge-View session is only between the edge-node and the user laptop (through the dispatcher). The controller is not involved in the data session path.

Dispatcher

This piece is new to the controller/edge-node model, although this can be part of the controller services at least initially from a security point of view. The UI needs to be flexible to generate the correct JWT token in various cases. The Golang example program for dispatcher will be published in 'github.com/lf-edge/eve/pkg/edgeview/dispatcher'. On a linux machine, do a 'make wss-server' in 'github.com/lf-edge/eve/pkg/edgeview' directory and copy the 'wss-server' image to the dispatcher server to run.

Dispatcher Location

This dispatcher is defined in JWT as the 'Dep' item. It has the IP address or domain name, the tcp port, and URL path to the Edge-View service. If this is controller based, it can be one of the implementations, and we need to decide which one:

- The service can use a different IP address than the IP of controller and use cloudflare to dispatch the traffic
- The service can use the same IP address as the IP of controller and listens on a different TCP port
- The service can use the same IP address as the IP of controller and the port 443 and use 'nginx' to multiplex the inbound traffic
- Have a different domain name controller.

The interaction of the service within controller can be simply a log file and some health monitoring service. The log can be included into fluentd and to be saved for search purpose.

Non Controller based Dispatcher

An enterprise customer may want to run and manage their own 'dispatcher' for some reason. For instance, if the edge-nodes and the laptop are both within their VPN domain, it may make sense to place the 'dispatcher' inside their VPN data center. Another use case can be due to Geo locations. If both the operator and edge-nodes are in Africa, they can place the dispatcher close to that region instead of hauling the traffic all the way to the US and back.

Data Path

There are three entities in the edge-view data operation, the user, the dispatcher and the edge-node. From TCP/TLS/Websocket connection POV, a user does not have any relation to the edge-node. The websocket connections are between the user with the dispatcher, and the edge-node with the dispatcher.

Think of this as the Hub-spoke model, with the dispatcher being the hub, and user and edge-node being two spokes. The user and the edge-node only have a 'virtual' connection which contains the 'application' layer packets, and the dispatcher is switching the packets for user and edge-node based on the hash value of a 'token'. This is analogous to the hub-spoke model in SD-WAN, where the hub installs the routes advertised from each spoke node, and based on the packet destination and VPN-index of the connection to do a lookup to forward packets to the right destination spoke. Here the dispatcher uses the 'token' similar to lookup for a VPN-ID to find the VPN-table. Since we only allow one user to interact with one edge-node (only two spokes within the same VPN), the hub only needs to find the 'other' spoke for the packet switching. In multiple instance case, the inst-ID is added to the hash of the token, thus it is still unique one to one mapping. This may change later for more complex topologies and use cases.

The JWT token has the nonce string, and it specifies the authentication or encryption choices. If authentication mechanism is used, all the messages between the laptop and EVE device through edge-view session will be authenticated through the Hash value calculated for the message and the nonce using HMAC Sha256 algorithm. If the encryption mechanism is used, all the messages will be encrypted by sha256 cryptographic algorithm using the nonce with the original message. Even if the 'dispatcher' is compromised on the path, it can not modify or decode the message since it does not know the nonce in the JWT token.

Status Upload

It is important for customers to easily determine if the Edge-View is being used on some of the edge-nodes within the enterprise and what type of access is being performed. The UI may need to offer different views from enterprise, project and device levels.

Status API

```
type EdgeviewStatus struct {
   ExpireOn    uint64    // unix time expiration in seconds
   StartedOn   time.Time // edge-view process started on timestamp
   CmdCountDev uint32    // total edge-view dev related commands performed
   CmdCountApp uint32    // total edge-view app related commands performed
}
```

For each edge-node currently is running Edge-View, it has the status of when it will expire, when the Edge-View started, total number of access so far, the last or current command string (those commands are not simple query for ‘route’ or ‘interface’, but ssh into the edge-node, VPC access into applications, TCP port access, etc), and the command type for those ‘well-known’ commands.

Three command types are defined above. Device, application, and external to the device. For any IP addresses access not belonging to the device, will be the external access type. E.g. http query for local profile server off the device.

The status will be uploaded as an event if the device is accessed with the ‘well-known’ commands (ssh type, reboot, vnc access, tcp port access, proxy). The CmdOption string is cleared after the command is done. The status will also be periodically updated if the ‘CmdCount’ value is changing, such as once every 10 minutes. The status will not be uploaded if the timestamp has expired.

Edgeview Commands to Device Events

The enterprise users will need to know what are the Edge-View debug commands being executed on the device and to the applications, and also where the sessions originated from. Each of the Edge-View commands received on the device will be logged and be sent to the controller. The session origination endpoint IP address(the public IP address) and port number are also included in the log message. Those log messages are tagged with a special object label.

On the controller when processing the device logs and finding this special object label, the content of the log message can be converted into a device event for the enterprise.

User View of Status

There should be an enterprise global view of the stats, such as the percentage of the total edge-nodes currently being provisioned with Edge-View. Users can directly go to an edge-node list and choose one device to see the details on the status. The device ‘Event’ tab page should include the Edge-View status events.

Edge-View Commands

The same docker edge-view container runs the EVE device can be run on the laptop for client side, assume the name is 'lfedge/eve-edgeview:snapshot' in lf-edge docker repo. An example of a docker run command on MacOS will be:

> *docker run -it --rm -h=$(hostname) -p 9001-9005:9001-9005 -v $HOME/tmp/download:/download lfedge/eve-edgeview:snapshot -token <JWT token> [-inst 1] <command>*

In the above example:

- this one is with instance number 1, in the multiple instances case
- port mapping 9001-9005 is used for TCP option in edge-view, every instance increases by 5 (9000 + instance# * 5)
- the volume maps your a local directory into the edge-view container /download. this will be used for file transfer, log files and techsupport files.

The edge-view commands can be seen through '-h':

> edge-view-query -token <session-token> [-debug] [-inst <instance-id>] <query string>
> options:
> log/search-pattern [-time <start_time>-<end_time> -json -type <app|dev> -line num]
>
> pub/ [baseosmgr domainmgr downloader edgeview global loguploader newlogd nim nodeagent tpmmgr vaultmgr volumemgr watcher zedagent zedclient zedmanager zedrouter]
>
> [acl app arp connectivity flow if mdns nslookup ping route socket speed tcp tcpdump trace url wireless]
> [app configitem cat cp datastore download du hw lastreboot ls model newlog pci ps cipher usb techsupport top volume]

To see more detail help, issue the command with '-h', for example 'app -h':

> ***edge-view-query.script app -h***
>
> app[/app-string] - to display all the app or one specific app
>
> e.g. app -- display all apps in brief
>
> app/iot -- display a specific app, which app name has substring of iot in more detail

Log Search

log search for string in the device log or application log. It can specify the time duration, either with a number(for hours) or with exact RFC3339 format. For example to search "panic:" in the log:

edge-view-query.script log/panic: -time 0-0.5 (for now back to half an hour before)

or in exact timestamp duration and output in json format:

edge-view-query.script log/panic: -time 2022-03-31T23:51:48Z-2022-03-31T23:45:00Z -json

Pub/Sub Info

those are all the pub/sub info the edge-view can query for:

[baseosmgr domainmgr downloader edgeview global loguploader newlogd nim nodeagent tpmmgr vaultmgr volumemgr watcher zedagent zedclient zedmanager zedrouter]

one can use comma to display multiple of them, e.g.:

edge-view-query.script pub/zedrouter,nim

one can also use '/' to give a substring of the topic, say from zedrouter and topic related to networkstatus:

edge-view-query.script pub/zedrouter/networkstatus

TCP Sessions

Edge-view TCP session command allows the user's laptop to be 'merged' into EVE's network for various TCP based operations. For instance, it can allow the user on the laptop to ssh into the EVE device, or ssh into one of the applications; The user can use VNC client to the console port of the VM/app or use browser for the HTTP services. e.g.

edge-view.script tcp/localhost:22/10.1.0.129:8080

the above TCP command allows the user to ssh (in another terminal) into the EVE device using port 9001, and also it allows the local browser to use 'http://localhost:9002' to the application HTTP service of one of the app on the device.

Other Commands

The detail description of other commands is out the scope of this document.

Edge-View PRs

API and JWT

https://github.com/lf-edge/eve/pull/2427

https://github.com/lf-edge/eve/pull/2558

Edge-View for EVE

https://github.com/lf-edge/eve/pull/2398